Aplicação das Bibliotecas Python para tratamento de dados em tempo real: A análise dos dados de isolamento social em Santa Catarina

Denis Vicentainer[1,a], Marcos Mattedi[1], Bruno Mello[1]

1 – Fundação Universidade Regional de Blumenau
a – Contato principal: denisvicentainer@gmail.com

Introdução

A necessidade controlar a crise causada pelo Covid-19 (Sars-Cov-2) fez emergir uma demanda, sem precedentes, por informações que possibilitam o monitoramento da propagação do vírus socialmente. Disponibilizar essa demanda de informação, em tempo real, exige dos pesquisadores e de seus sistemas de gerenciamento de dados maior celeridade. Os dados epidemiológicos são a principal referência quanto a eficiência das medidas controladoras implementadas, influenciando diretamente não apenas os gestores públicos para a tomada de decisão, mas também a sociedade para a prevenção. Uma problemática extremamente comum, que ocorre em eventos como epidemias e pandemias, é a de gerenciamento dos dados. Esse problema ainda é agravado por gestões fragmentadas que utilizam diferentes métodos de coleta, de análise, de tratamento e de ferramentas digitais, expondo um problema organizacional.

Os municípios de estado de Santa Catarina registram pouca efetividade para a gestão dos dados sobre a pandemia. Muitos municípios simplesmente não têm informações, outros apresentam informações incompletas, sendo poucos os que apresentam dados consistentes. Além disso, muitos municípios utilizam diferentes padrões de armazenamento, de análise e de tratamento, o que dificulta a manipulação para o agrupamento em uma escala regional ou estadual. Nesse contexto, o problema de pesquisa recai sobre a dificuldade de mapear dados em escala estadual, a partir de dois fenômenos: **a) demanda excessiva de informação sobre a pandemia e; b) a gestão fragmentada desses dados, no qual municípios utilizam diferentes padrões, plataformas, métodos de análise e tratamento etc.** Com isso, algumas ferramentas tradicionais de manipulação de dados[1] não atendem à demanda e tornam o processo de análise, tratamento e mapeamento dos dados menos eficiente.

Neste sentido, as linguagens de programação[2] podem contribuir para tornar esse processo mais ágil e assertivo. As linguagens de programação podem reduzir o fluxo de trabalho com a automatização do tratamento de dados em poucas linhas de código[3]. As linguagens podem fornecer uma construção interativa que auxilia na pré-visualização dos processos aplicados sobre o dado, além da agilidade em extrair estatísticas, manipular arquivos e realizar o mapeamento categorizado. Dentre as diversas linguagens de programação, o *Python*[4], tem se notabilizado por sua variedade de bibliotecas[5] direcionadas para as áreas de ciência de dados, aumentando sua eficiência enquanto ferramenta de tratamento e análise de dados e estatísticas.

[1] Entre algumas das ferramentas mais comuns para o gerenciamento e manipulação de dados, estão os *softwares* excel e libreoffice para elementos matriciais e Arcgis e o Qgis para dados georreferenciados.
[2] Linguagem de programação: instruções de processamento ao computador.
[3] Linhas de código: código fonte para execução de programas.
[4] Phyton: sistema de linguagem de programação de alto nível.
[5] Bibliotecas: Conjunto de módulos e funções.

10.21166/metapre.v3i0.1392 ISSN: 2674-9009

Nesse contexto, a pergunta a se responder nesse artigo é: **como o uso da linguagem de programação *Python* pode contribuir para maior celeridade nos processos de análise, tratamento e visualização de dados referentes ao Covid-19 em Santa Catarina?**

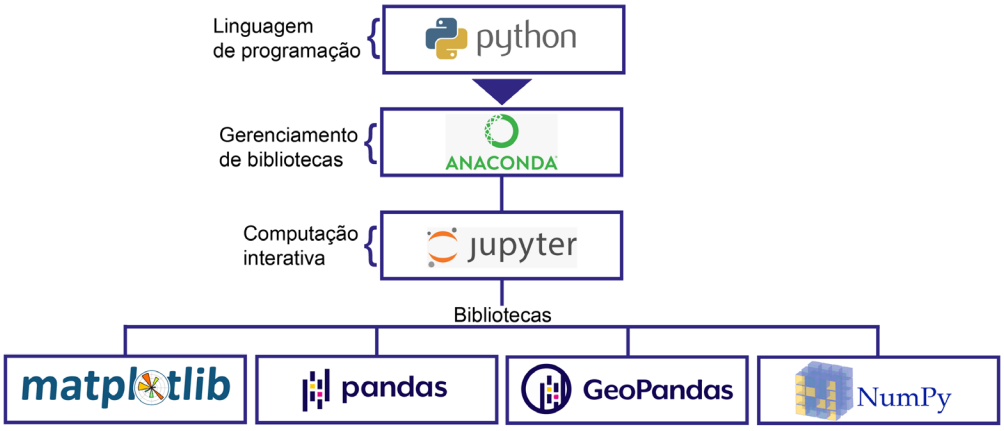
Para tanto, este artigo objetiva organizar um método de análise, tratamento e compatibilização de dados por meio de ferramentas disponíveis em linguagem *Python*. A metodologia fica explicitada por uma pesquisa descritiva/explicativa dos processos de tratamento de dados de isolamento social no Estado de Santa Catarina entre fevereiro e maio de 2020. O texto está dividido em três partes, nas quais: i) pressupostos operacionais e instalação de bibliotecas; ii) tratamento e análise estatística do dado de isolamento social em Santa Catarina e; iii) elaboração do mapeamento. Este artigo vem no sentido de demonstrar a importância da ciência de computação, mais especificamente da programação, para o processo de análise, tratamento e mapeamento de dados, reforçando seu caráter interdisciplinar.

1 – Pressupostos operacionais e instalação de bibliotecas

1.1 - Identificação das ferramentas disponíveis

Existe atualmente a disponibilidade de muitas ferramentas de tratamento de informação. Assim, um primeiro desafio para o monitoramento de dados em tempo real compreende a seleção e contabilização do mix de ferramentas. Para a preparação do ambiente de trabalho faremos uso da plataforma *open-source*[6] de gerenciamento de bibliotecas, *Anaconda*, incluindo a instalação do ambiente computacional web para a escrita de códigos interativos, *Jupyter Notebook*. As bibliotecas utilizadas neste projeto serão: *pandas*, para manipulação de matrizes; *matplotlib*, para realizar a plotagem de gráficos; *numpy*, para auxiliar a conversão entre tipos de dados; e *geopandas*, para importação de dados georreferenciados dentro do ambiente de trabalho (Figura 1).

Figura 1 – Diagrama relacional das ferramentas do ambiente de trabalho



Fonte: Autores, 2020.

[6] Open-source: software de código aberto.

10.21166/metapre.v3i0.1392 ISSN: 2674-9009

1.2 - Preparando o ambiente de trabalho

Embora grande parte das bibliotecas utilizadas façam parte da instalação padrão do Anaconda, o uso de bibliotecas externas pode ocasionar conflitos entre versões da instalação padrão e aquelas adicionadas posteriormente. Deste modo, o projeto é iniciado com a criação de um ambiente virtual isolado, onde realiza-se a instalação das bibliotecas em suas versões específicas de forma independente. O processo inicial é realizado dentro do *Anaconda prompt*[7], da seguinte forma:

```
>>> conda create --name env python=3
```

O código acima cria o ambiente virtual, onde 'env' corresponde ao nome do ambiente e 'python=3' à versão da linguagem *Python* utilizada. O ambiente é ativado com o código:

```
>>> conda activate env
```

Com o novo ambiente ativado, as livrarias serão instaladas de forma independente, porém, é necessário garantir que sejam acessadas através do Jupyter Notebook, permitindo o desenvolvimento interativo da análise. Para isto, utiliza-se o pacote *ipykernel*[8]*:*

```
>>> pip install ipykernel
```

```
>>> python -m ipykernel install --user name env --display-name "iso_covid"
```

Assim, realiza-se a integração entre o Jupyter Notebook e o ambiente virtual, "iso_covid" aparecerá como ambiente virtual disponível para a criação de um novo *notebook*[9]. Na sequência é a realizada a instalação das bibliotecas:

```
>>>conda install -n env numpy
```

```
>>>conda install -n env maplotlib
```

```
>>>conda install -n env pandas
```

```
>>>conda install -n env geopandas
```

Terminada a instalação, as bibliotecas devem ser importadas para plataforma Jupyter Notebook:

```
>>>import geopandas as gpd
```

```
>>>import pandas as pd
```

```
>>>import matplotlib.pyplot as plt
```

```
>>>import numpy as np
```

2 – Limpeza e análise do dado de isolamento social em Santa Catarina

2.2 - Carregando dados para o ambiente de trabalho

Para a análise e tratamento de dados utilizou-se a base de dados de isolamento social do Estado de Santa Catarina em formato *csv*[10] gentilmente disponibilizados pelo Prof. Dr. Pedro

[7] Anaconda Prompt: interface de linhas de comando.
[8] Ipykernel: Ipython Kernel é o backend de execução Python para Jupyter Notebook.
[9] Notebook: interface de desenvolvimento integrado do Jupyter Notebook.
[10] Formato csv: arquivo de valores separados por vírgulas.

10.21166/metapre.v3i0.1392 ISSN: 2674-9009

da Silva Peixoto do Departamento de Matemática Aplicada do Instituto de Matemática e Estatística da USP, e o arquivo *shapefile*[11], coletado da base de dados do IBGE (2019), informando os limites territoriais dos municípios de Santa Catarina. Primeiro, carrega-se o dado matriz em formato *csv* para o Jupyter Notebook chamando a função '*pd.read_csv'* da biblioteca *pandas* que possibilita sua conversão para objeto *dataframe*[12] e atribuí-lo à variável 'df':

```
>>>df = pd.read_csv('caminho\para\o\arquivo\iso_data.csv', sep=';')

>>>df
```

Uma vez carregadas, examinam-se as informações contidas no arquivo chamando a variável '*df',* que retorna a Figura 2. A plataforma Jupyter Notebook executa cada linha de código interativamente, plotando o resultado da operação. Neste caso, a chamada da variável '*df*' retorna, por padrão, as cinco primeiras e cinco últimas linhas da matriz, permitindo inspecionar, de modo geral, os nomes das colunas, a natureza dos valores contidos nas linhas e a relação entre quantidade de linhas e colunas.

Figura 2 – Resultado da chamada do *dataframe* df

	Unnamed: 0	state_name	reg_name	iso	day	state_abrv	reg_state	COD
0	450425	Santa Catarina	ÁGUA DOCE	0.254902	04/02/2020	SC	ÁGUA DOCE_SC	1.0
1	324850	Santa Catarina	ÁGUA DOCE	0.269231	06/03/2020	SC	ÁGUA DOCE_SC	1.0
2	310255	Santa Catarina	ÁGUA DOCE	0.280000	10/03/2020	SC	ÁGUA DOCE_SC	1.0
3	67709	Santa Catarina	ÁGUA DOCE	0.318841	08/05/2020	SC	ÁGUA DOCE_SC	1.0
4	42996	Santa Catarina	ÁGUA DOCE	0.337838	14/05/2020	SC	ÁGUA DOCE_SC	1.0
...	...	...	...	...	...	...	...	...
19546	254207	Santa Catarina	XAXIM	0.579251	24/03/2020	SC	XAXIM_SC	NaN
19547	236031	Santa Catarina	XAXIM	0.599462	29/03/2020	SC	XAXIM_SC	NaN
19548	264264	Santa Catarina	XAXIM	0.711957	22/03/2020	SC	XAXIM_SC	NaN
19549	237312	Santa Catarina	ZORTÉA	0.470588	28/03/2020	SC	ZORTÉA_SC	NaN
19550	101184	Santa Catarina	ZORTÉA	0.518519	01/05/2020	SC	ZORTÉA_SC	NaN

19551 rows × 8 columns

Fonte: Autores, 2020.

2.3 – Inspecionando o dado

Tratando-se de um dado referente ao isolamento social diário por município, identificam-se as colunas com relevância de informação para análise: '*reg_name'*, contendo o nome dos municípios, '*iso*' contendo a porcentagem do isolamento social diária e '*day*' contendo o dia da medição do isolamento. Chamando a função '*.info()'* verifica-se o tipo de dado contido em cada coluna, conforme Figura 3.

```
>>> df.info()
```

[11] Shapefile: formato de arquivo contendo dados geoespaciais.
[12] Dataframe: dado tabular bidimensional.

10.21166/metapre.v3i0.1392 ISSN: 2674-9009

Figura 3 – Resultado retornado por *df.info()* (destacadas colunas e seus tipos de valores)

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 19551 entries, 0 to 19550
Data columns (total 8 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Unnamed: 0  19551 non-null  int64
 1   state_name  19551 non-null  object
 2   reg_name    19551 non-null  object
 3   iso         19551 non-null  float64
 4   day         19551 non-null  object
 5   state_abrv  19551 non-null  object
 6   reg_state   19551 non-null  object
 7   COD         4598 non-null   float64
dtypes: float64(2), int64(1), object(5)
memory usage: 1.2+ MB
```

Fonte: Autores, 2020.

A visualização das informações presentes na matriz e o tipo de dado de cada uma de suas colunas indicam a necessidade de quatro etapas para sua limpeza e preparação:

a) exclusão das colunas excedentes;

```
>>>df.drop(['Unnamed:0', 'state_name', 'reg_state', 'COD'],1, inplace=True)
```

b) remoção dos caracteres de acentuação presentes na coluna ‘reg_name’;

```
>>>cols = df.select_dtypes(include=[np.object]).columns

>>> df[cols] = df[cols].apply(lambda x: x.str.normalize('NFKD').str.encode('ascii',
errors='ignore').str.decode('utf-8'))
```

c) conversão do tipo de dado da coluna ‘day’ para datetime;

```
>>>df['day'] = pd.to_datetime(df['day'], dayfirst=True)
```

d) renomear colunas essenciais, facilitando a localização das informações.

```
>>>df.rename(columns = {'reg_name':'NM_MUN', 'iso':'ISO', 'day':'DATA',
'state_abrv':'SIGLA_UF'}, inplace=True)
```

O processo de limpeza e padronização do dado permitirá controlar a aplicação de operações e unificá-lo com dados advindos de outras bases. De acordo com a Figura 4, a antiga coluna ‘*day*’ renomeada para ‘*DATA*’ com seus valores convertidos para *datetime* permite a classificação simultânea da matriz por nome do Município em ordem alfabética e data da coleta do percentual de isolamento, chamando a função ‘*.sort_values()*’ sobre a variável ‘*df*’:

```
>>>df.sort_values(by=['NM_MUN', 'DATA'], ascending=[True, True],inplace=True)

>>>df
```


10.21166/metapre.v3i0.1392 ISSN: 2674-9009

Figura 4 – *Dataframe* 'df' retornado após limpeza e classificação

	NM_MUN	ISO	DATA	SIGLA_UF
202	ABELARDO LUZ	0.348148	2020-02-01	SC
241	ABELARDO LUZ	0.405172	2020-02-02	SC
205	ABELARDO LUZ	0.352459	2020-02-03	SC
200	ABELARDO LUZ	0.343511	2020-02-04	SC
195	ABELARDO LUZ	0.330709	2020-02-05	SC
...	...	...	...	...
19491	XAXIM	0.367292	2020-05-21	SC
19501	XAXIM	0.390710	2020-05-22	SC
19531	XAXIM	0.473418	2020-05-23	SC
19549	ZORTEA	0.470588	2020-03-28	SC
19550	ZORTEA	0.518519	2020-05-01	SC

19551 rows × 4 columns

Fonte: Autores, 2020.

Neste ponto o *dataframe* apresenta as informações necessárias, facilitando o entendimento do conteúdo e interpretação dos valores. A coluna '*ISO*', contendo o percentual de isolamento é a informação mais relevante para o desenvolvimento da análise. Utilizando-a como indexadora é possível gerar descrições estatísticas e compreender a amplitude dos valores contidos na coluna chamando a função '*.describe()*', que retornará ('*count*', somatório de linhas contendo valores; '*mean*', media dos valores; '*std*', desvio padrão dos valores; 'min', valor mínimo; os percentis '25%', '50%' e '75%' e; 'max', valor máximo)(Figura 5):

```
>>>df['ISO'].describe()
```

Figura 5 – Resultado retornado por df['ISO'].describe()

```
count    19551.000000
mean         0.399874
std          0.100482
min          0.111111
25%          0.326004
50%          0.391768
75%          0.456941
max          0.854545
Name: ISO, dtype: float64
```

Fonte: Autores, 2020.

10.21166/metapre.v3i0.1392 ISSN: 2674-9009

2.4 – Trabalhando com grupo de valores e gráficos estatísticos

A biblioteca *pandas* proporciona o agrupamento de dados por valores das colunas, esta operação possibilita filtrar dados individuais por nome do Município:

```
>>>reg_grp0 = df.groupby(['NM_MUN'])
```

Cria-se o grupo de valores (objeto tipo '*DataFrameGroupBy*[13]') chamando a função *'.groupby()'*, armazenando-a na variável '*reg_grp0'* e passando a coluna '*NM_MUN'* como parâmetro. No processo seguinte é chamada a função *'.get_group()*' sobre a variável '*reg_grp0'* com o nome do Munícipio desejado passado como parâmetro, neste exemplo, *'FLORIANOPOLIS*'(Figura 6):

```
>>>reg_grp0.get_group('FLORIANOPOLIS')
```

Figura 6 – Dados filtrados por nome de Município

	NM_MUN	ISO	DATA	SIGLA_UF
5726	FLORIANOPOLIS	0.312231	2020-02-01	SC
5736	FLORIANOPOLIS	0.388723	2020-02-02	SC
5708	FLORIANOPOLIS	0.264038	2020-02-03	SC
5725	FLORIANOPOLIS	0.306462	2020-02-04	SC
5712	FLORIANOPOLIS	0.278904	2020-02-05	SC
...	...	...	...	...
5747	FLORIANOPOLIS	0.427512	2020-05-19	SC
5744	FLORIANOPOLIS	0.421741	2020-05-20	SC
5748	FLORIANOPOLIS	0.428856	2020-05-21	SC
5750	FLORIANOPOLIS	0.435795	2020-05-22	SC
5762	FLORIANOPOLIS	0.455555	2020-05-23	SC

113 rows × 4 columns

Fonte: Autores, 2020.

A criação do grupo '*reg_grp0'* é de extrema importância. Por um lado, permite a criação de uma nova matriz de valores baseada nos percentis de cada município. Por outro, viabiliza a utilização de funções da biblioteca '*matplotlib*' para extração de gráficos estatísticos que demonstram a evolução do isolamento social por cidade durante o período coletado. Armazena-se o grupo, filtrado pelo Município de Florianópolis, na variável '*flp_df*', na linha seguinte, chama-se a função '*plt.plot()*' onde os atributos de '*flp_df*' são passados como parâmetros '*flp_df.DATA'* e '*flp_df.ISO'* sendo plotados na orientação x e y, respectivamente, do gráfico disposto na Figura 7:

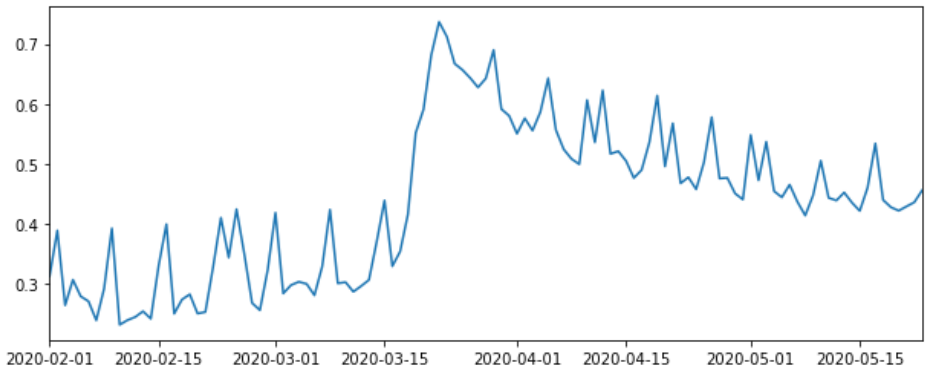
```
>>>flp_df = reg_grp0.get_group('FLORIANOPOLIS')

>>>plt.plot(flp_df.DATA, flp_df.ISO)
```

[13] DataFrameGroupBy: dado tabular bidimensional agrupado por valores.

10.21166/metapre.v3i0.1392 ISSN: 2674-9009

```
>>>plt.show
```

Figura 7 – Evolução do isolamento social em Florianópolis



Fonte: Autores, 2020.

O gráfico evidencia que Florianópolis registrou seu pico de isolamento durante os dias 15 de março e 01 de abril de 2020. As funções para plotagem de gráficos da biblioteca '*matplotlib*' são úteis para compreender visualmente o comportamento do dado analisado. Nota-se que o dado é estruturado para acompanhar o mapeamento diário do isolamento das cidades. Entretanto, é possível gerar índices estatísticos que possibilitem aferir acerca das diferenças entre os percentuais de isolamento, durante todo o período coletado, chamando a função '*.agg()'* sobre o objeto de grupo de valores '*reg_grp0'* e passando como parâmetro uma lista para criação de índices estatísticos (Figura 8), armazenados na variável 'iso_estats':

```
>>>iso_estats = reg_grp0['ISO'].agg(['median', 'mean', 'max', 'min'])

>>>iso_estats
```

Figura 8 – Índices estatísticos

NM_MUN	median	mean	max	min
ABELARDO LUZ	0.380952	0.384876	0.550847	0.210526
AGROLANDIA	0.393443	0.400774	0.664062	0.250000
AGRONOMICA	0.429563	0.431683	0.854545	0.192308
AGUA DOCE	0.426471	0.428035	0.580645	0.254902
AGUAS DE CHAPECO	0.420000	0.438014	0.490197	0.403846
...	...	...	...	...
VITOR MEIRELES	0.476364	0.462569	0.568627	0.333333
WITMARSUM	0.529412	0.529412	0.666667	0.392157
XANXERE	0.360791	0.364384	0.636842	0.219269
XAXIM	0.368700	0.382004	0.711957	0.238854
ZORTEA	0.494553	0.494553	0.518519	0.470588

213 rows × 4 columns

Fonte: Autores, 2020.

10.21166/metapre.v3i0.1392 ISSN: 2674-9009

As médias de isolamento '*median*' serão utilizadas para mapear comparativamente a média de isolamento entre municípios durante o período coletado. Contudo, varia por cidade a quantidade de dias com registro de isolamento. Deste modo, serão extraídos do *dataframe 'df'* para o dicionário '*mais_2meses'* apenas municípios com mais que 60 dias coletados, considerados suficientemente consistentes para mapeamento:

```
>>>mais_2meses = {}

>>>for value, count in df.NM_MUN.value_counts().iteritems():

    if count > 60:

        mais_2meses[value] = count

>>>df_mais_2meses = pd.DataFrame(mais_2meses.items(), columns=['NM_MUN', 'COUNT'])
```

O *dataframe* '*df_mais_2meses'* é criado com as colunas '*NM_MUN'* com valores dos nomes dos municípios e '*COUNT*' com a somatória dos dias registrados por município. Nele, serão unidos os índices estatísticos de '*iso_estats'*, utilizando como identificador a coluna *'NM_MUN'* e atribuídos à variável 'sc_mais_2meses'(Figura 9):

```
>>>sc_mais_2meses = pd.merge(iso_estats, df_mais_2meses, on='NM_MUN')

>>>sc_mais_2meses
```

Figura 9 – Resultado da chamada da variável 'sc_mais_2meses' (índices estatísticos unidos com o somatório de dias registrado por município)

	NM_MUN	median	mean	max	min	COUNT
0	ABELARDO LUZ	0.380952	0.384876	0.550847	0.210526	113
1	AGROLANDIA	0.393443	0.400774	0.664062	0.250000	113
2	AGRONOMICA	0.429563	0.431683	0.854545	0.192308	82
3	AGUAS MORNAS	0.405660	0.413192	0.708738	0.222749	113
4	ALFREDO WAGNER	0.360825	0.354106	0.569892	0.170000	113
...	...	...	...	...	...	...
163	VARGEM BONITA	0.419355	0.435557	0.716981	0.254545	101
164	VIDAL RAMOS	0.390805	0.402287	0.653846	0.274510	65
165	VIDEIRA	0.404101	0.415134	0.722081	0.267788	113
166	XANXERE	0.360791	0.364384	0.636842	0.219269	113
167	XAXIM	0.368700	0.382004	0.711957	0.238854	113

168 rows × 6 columns

3 – Elaboração do mapeamento

Para realizar o mapeamento de dados georreferenciados, utiliza-se a biblioteca '*geopandas*'. Através dela é importado o arquivo *shapefile* com os limites territoriais dos municípios de Santa Catarina (IBGE 2019)(Figura 9), chamando a função '*.read_file()*' para carregar o arquivo e atribuí-lo à variável '*sc_data'*:

10.21166/metapre.v3i0.1392 ISSN: 2674-9009

```
>>>sc_data = gpd.read_file('Caminho\para\o\arquivo\SC_Municipios_2019.shp')

>>>sc_data
```

Figura 10 – Dados do IBGE convertidos em *dataframe*

	CD_MUN	NM_MUN	SIGLA_UF	AREA_KM2	geometry
0	4200051	Abdon Batista	SC	237.517	POLYGON ((-51.12356 -27.62134, -51.12387 -27.6...
1	4200101	Abelardo Luz	SC	953.992	POLYGON ((-52.41878 -26.58662, -52.42002 -26.5...
2	4200200	Agrolândia	SC	206.815	POLYGON ((-49.81594 -27.35622, -49.80909 -27.3...
3	4200309	Agronômica	SC	129.774	POLYGON ((-49.69655 -27.25605, -49.69640 -27.2...
4	4200408	Água Doce	SC	1319.137	POLYGON ((-51.69250 -26.80284, -51.69255 -26.8...
...	...	...	...	...	...
290	4219507	Xanxerê	SC	377.426	POLYGON ((-52.46637 -26.89232, -52.47047 -26.8...
291	4219606	Xavantina	SC	218.032	POLYGON ((-52.29672 -26.96118, -52.29666 -26.9...
292	4219705	Xaxim	SC	293.628	POLYGON ((-52.61777 -26.90850, -52.61636 -26.9...
293	4219853	Zortéa	SC	190.179	POLYGON ((-51.55718 -27.45336, -51.55723 -27.4...
294	4220000	Balneário Rincão	SC	63.420	POLYGON ((-49.29771 -28.84144, -49.29776 -28.8...

295 rows × 5 columns

Fonte: IBGE, 2019.

A biblioteca *geopandas* utiliza métodos e funções similares aos da biblioteca '*pandas*'. Igualmente, a chamada da variável '*sc_data'* retorna o nome das colunas, os valores de suas cinco primeiras e cinco últimas linhas e uma relação entre número de linhas e colunas. As colunas de maior relevância para este processo serão, '*NM MUN'* contendo o nome dos municípios e '*geometry*' com polígonos georreferenciados de seus limites. A coluna '*MN_MUN'* será utilizada como identificador para unir os *dataframes* armazenadas em '*sc_data'* e 'sc_mais_2meses'. Portanto, é necessário a realização do processo de padronização, executando a limpeza de caracteres de acentuação e sua transformação para letras maiúsculas:

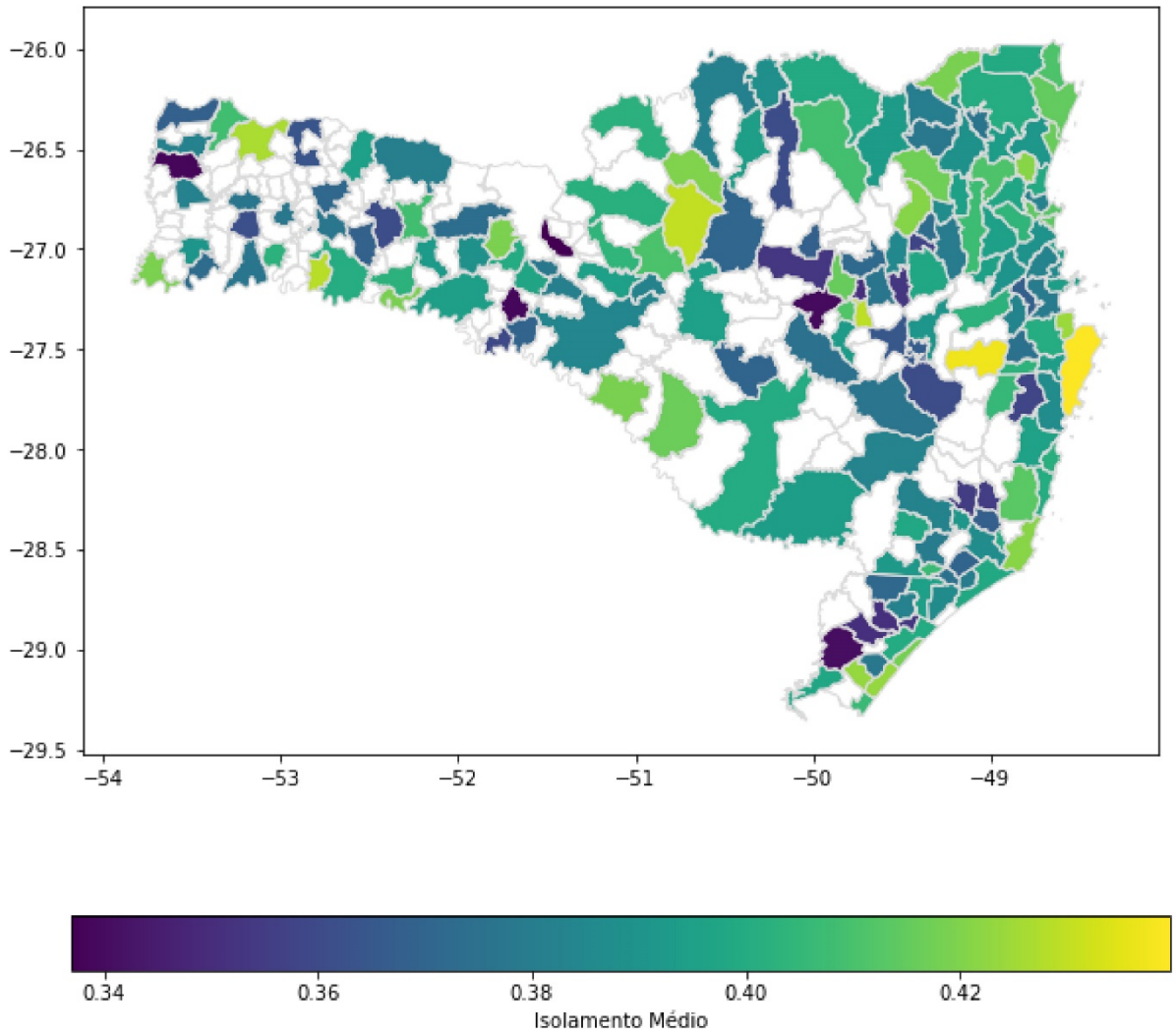
```
>>>cols = sc_data.select_dtypes(include=[np.object]).columns

>>>sc_data[cols] = sc_data[cols].apply(lambda x:
x.str.normalize('NFKD').str.encode('ascii', errors='ignore').str.decode('utf-8'))

sc_data['NM_MUN'] = sc_data['NM_MUN'].str.upper()
```

A união entre sc_data e sc_mais_2meses permite que os valores médios das cidades com mais de 60 dias de registro do isolamento social sejam mapeados e exibidos, conforme Figura 10:

```
>>>ax = sc_data.plot(figsize=(10,10), color='none', edgecolor='gainsboro',
zorder=3)

>>>sc_mais_2meses.plot(column='median', ax=ax, legend=True, legend_kwds={'label':
'Isolamento Médio', 'orientation':'horizontal'})
```


10.21166/metapre.v3i0.1392 ISSN: 2674-9009

Figura 11 – Média de isolamento por município entre 01/02/2020 e 15/05/2020



Fonte: Autores, 2020.

Conclusão

Um fenômeno da amplitude da pandemia causada pela Covid-19 ocasionou um excesso de informações, advindas de bases variadas com distintas metodologias e formatos. Nesse contexto, os *softwares* mais comuns para manipulação de dados necessitam realizar operações mais complexas para o tratamento e padronização dos dados de origem desconhecida, tornando o processo lento. Por sua vez, a linguagem *Python,* tem se notabilizado pela variedade de bibliotecas que facilitam a execução de processos essenciais para o gerenciamento do ciclo de vida dos dados. Ou seja, o *Phyton* pode permitir aos pesquisadores maior agilidade e liberdade na inspeção de dados, pois oferecem uma interface que possibilita trabalhar com os mais variados formatos de arquivos de forma dinâmica.

O excesso de informações e uma gestão fragmentada dos dados tornam o processo de manipulação de dados ineficientes em *softwares* tradicionais. Nesse sentido, foram realizados procedimentos que comprovam a eficiência da linguagem de programação *Python* utilizando como referência os dados de isolamento social no estado de Santa Catarina. Respondendo à

10.21166/metapre.v3i0.1392 ISSN: 2674-9009

pergunta de partida, o *Python* permitiu executar a limpeza de caracteres indesejados, remoção de colunas excedentes, a conversão entre diferentes tipos de valores nos dados de isolamento social dos municípios e um mapeamento dos dados disponibilizadas em diferentes plataformas e métodos. Isso comprova que quaisquer dados digitais, em diferentes formatos podem ser padronizados com maior celeridade e assertividade no *Phython*. Porém, ainda a há muito que se avançar em relação a gestão dos dados no estado de Santa Catarina, principalmente quanto a coleta, padronização, armazenamento e acesso aos dados.

REFERÊNCIAS

ANACONDA. Guia de uso do Anaconda. Disponível em: https://docs.anaconda.com/anaconda/user-guide/getting-started/ > Acesso: 13 set. 2020

GEOPANDAS. Documentação operacional da biblioteca. Disponível em: https://geopandas.org/ Acesso: 02 ago. 2020.

IBGE – Instituto Brasileiro de Geografia e Estatística. Cidades. 2019. Disponível em: https://www.ibge.gov.br/geociencias/downloads-geociencias.html Acesso: 30 jun. 2020.

IPYKERNEL. Documentação operacional da biblioteca. Disponível em:https://ipython.readthedocs.io/en/stable/install/kernel_install.html Acesso: 08 ago. 2020.

MATPLOTLIB. Documentação da biblioteca. Disponível em: <https://matplotlib.org/api/pyplot_summary.html> Acesso: 27 ago. 2020.

NUMPY. Documentação da biblioteca. Disponível em: https://docs.scipy.org/doc/ Acesso: 13 set. 2020.

PANDAS. Documentação operacional da biblioteca. Disponível em: https://pandas.pydata.org/docs/ Acesso: 15 jul. 2020.

PYTHON. Documentação da linguagem Python Disponível em: https://docs.python.org/pt-br/3/ Acesso: 13 set. 2020.

10.21166/metapre.v3i0.1392 ISSN: 2674-9009